**Assignment 27.1**

**Problem Statement:**

Explain the following in brief with an example.

● Map side Join

● Reduce side Join

● Bucket Map Join

● SMBM Join

**Answer:**

* **Map side Join:**

In case one of the dataset is small, map side join takes place.

Map-side joins can be enabled in two ways.

1. By specifying the keyword, /\*+ MAPJOIN(b) \*/ in the join statement.
2. By setting the following property to true.

hive.auto.convert.join=true

For performing Map-side joins, there should be two files, one is of larger size and the other is of smaller size.

You can set the small file size by using the following property:

hive.mapjoin.smalltable.filesize=(default it will be 25MB)

In map side join, a local job runs to create hash-table from content of HDFS file and sends it to every node.

set hive.auto.convert.join=true;

***For Example,***

SELECT /\*+ MAPJOIN(dataset2) \*/ dataset1.first\_name, dataset1.eid,dataset2.eid FROM dataset1 JOIN dataset2 ON dataset1.first\_name = dataset2.first\_name;

* **Reduce side Join:**

Reduce-Side joins are simpler than Map-Side joins since the input datasets need not to be structured. But it is less efficient as both datasets have to go through the MapReduce shuffle phase.

The records with the same key are brought together in the reducer.

We can also use the Secondary Sort technique to control the order of the records.

If datasets are large, reduce side join takes place**.**

* **Bucket Map Join:**

The data must be bucketed on the keys used in the ON clause and the number of buckets for one table must be a multiple of the number of buckets for the other table.

The constraint for performing Bucket-Map join is:

If tables being joined are bucketed on the join columns, and the number of buckets in one table is a multiple of the number of buckets in the other table, the buckets can be joined with each other.

To perform bucketing, we need to have bucketed tables

When these conditions are met, Hive can join individual buckets between tables in the map phase, because it does not have to fetch the entire content of one table to match against each bucket in the other table.

set hive.optimize.bucketmapjoin=true;

set hive.auto.convert.join=true;

***For Example,***

CREATE TABLE IF NOT EXISTS dataset1\_bucketed (eid int,first\_name String, last\_name String, email String, gender String, ip\_address String) clustered by(first\_name) into 4 buckets row format delimited fields terminated BY ',';

CREATE TABLE IF NOT EXISTS dataset2\_bucketed (eid int,first\_name String, last\_name String) clustered by(first\_name) into 8 buckets row format delimited fields terminated BY ',' ;

insert into dataset1\_bucketed select \* from dataset1;

insert into dataset2\_bucketed select \* from dataset2;

SELECT /\*+ MAPJOIN(dataset2\_bucketed) \*/ dataset1\_bucketed.first\_name,dataset1\_bucketed.eid, dataset2\_bucketed.eid FROM dataset1\_bucketed JOIN dataset2\_bucketed ON dataset1\_bucketed.first\_name = dataset2\_bucketed.first\_name ;

* **SMBM Join:**

Sort-Merge-Bucket (SMB) joins can be converted to SMB map joins as well.

SMB joins are used wherever the tables are sorted and bucketed.

If the tables being joined are sorted and bucketized on the join columns and have the same number of buckets, a sort-merge join can be performed. The corresponding buckets are joined with each other at the mapper.

For performing the SMB-Map join, we need to set the following properties:

Set hive.input.format=org.apache.hadoop.hive.ql.io.BucketizedHiveInputFormat;

set hive.optimize.bucketmapjoin = true;

set hive.optimize.bucketmapjoin.sortedmerge = true;

To perform this join, we need to have the data in the bucketed tables sorted by the join column.

The join boils down to just merging the already sorted tables, allowing this operation to be faster than an ordinary map-join.

set hive.enforce.sortmergebucketmapjoin=false;

set hive.auto.convert.sortmerge.join=true;

set hive.optimize.bucketmapjoin = true;

set hive.optimize.bucketmapjoin.sortedmerge = true;

***For Example,***

insert overwrite table dataset1\_bucketed select \* from dataset1 sort by first\_name;

insert overwrite table dataset2\_bucketed select \* from dataset2 sort by first\_name;

SELECT /\*+ MAPJOIN(dataset2\_bucketed) \*/ dataset1\_bucketed.first\_name,dataset1\_bucketed.eid, dataset2\_bucketed.eid FROM dataset1\_bucketed JOIN dataset2\_bucketed ON dataset1\_bucketed.first\_name = dataset2\_bucketed.first\_name ;

CREATE TABLE IF NOT EXISTS dataset2\_bucketed1 (eid int,first\_name String, last\_name String) clustered by(first\_name) into 4 buckets row format delimited fields terminated BY ',' ;

insert overwrite table dataset2\_bucketed1 select \* from dataset2 sort by first\_name;

SELECT /\*+ MAPJOIN(dataset2\_sbucketed1) \*/dataset1\_bucketed.first\_name, dataset1\_bucketed.eid, dataset2\_bucketed1.eid FROM dataset1\_bucketed JOIN dataset2\_bucketed1 ON dataset1\_bucketed.first\_name = dataset2\_bucketed1.first\_name ;